Software development cultures and cooperation problems: a field study of the early stages of development of software for a scientific community

Journal Item

Software development cultures and cooperation problems: a field study of the early stages of development of software for a scientific community

Judith Segal

Empirical Studies of Software Development Group

Centre for Research in Computing

The Department of Computing

The Open University

Milton Keynes MK7 6AA

UK

j.a.segal@open.ac.uk

phone: +44 (0)1908 659793

fax: +44 (0) 1908 652335

Development cultures and cooperation problems

Abstract

In earlier work, I identified a particular class of end-user developers, who include scientists and whom I term 'professional end-user developers', as being of especial interest. Here, I extend this work by articulating a culture of professional end-user development, and illustrating by means of a field-study how the influence of this culture causes cooperation problems in an inter-disciplinary team developing a software system for a scientific community. My analysis of the field study data is informed by some recent literature on multi-national work cultures. Whilst acknowledging that viewing a scientific development through a lens of software development culture does not give a full picture, I argue that it nonetheless provides deep insights.

Keywords

Community software development; cooperation; field study; scientific software development; software development culture; professional end-user developers

1. Introduction

The focus of this paper is on the influence of a particular software development culture, a professional end-user development culture, on problems of cooperation. The paper is based on an ongoing field study and describes those cooperation problems which arose in the early stages of development of some infrastructure software for the structural protein community. This software is a **l**aboratory **i**nformation **m**anagement **s**ystem, a LIMS, which consists essentially of a web-based application back-ended by a database (see 3.1 below for more details). I shall refer to this system as FSLIMS (Field Study Laboratory Information Management System). There are essentially four different groups of stakeholders involved in this development, and cooperation between and within these different groups is essential for a successful development. This cooperation was, however, beset by problems. Many of these can be explained, in part at least, in terms of incongruences between the values espoused by the scientists in the user community, who were steeped in the professional end-user development culture, and those of the project manager of the development, who is a software engineer.

The structure of this paper is as follows. In section 2, I describe a model, based on my previous field studies, of the culture (values and customary behaviours) of scientists developing software for themselves. Such scientists may be described as 'professional end-user developers', that is, people who do not consider themselves primarily as software developers, but who work in highly technical, knowledge-rich domains and who develop software in order to further their own professional goals, Segal (2007). They are distinguished from other end-user developers in that coding per se presents few problems to them as they are used to abstractions and formal languages. In section 3, I set the scene for the field study. I describe (briefly) the biology and the lab practice that FSLIMS is intended to support, the main groups of stakeholders in the development, the role of collaboration within and between these groups, and the methodology of the study. In section 4, I describe the major cooperation problems which arose in the early stages of development, discuss some possible causes, and describe how the problems were addressed so as to enable the development to proceed. In section 5, I briefly describe some recent literature on frameworks for analysing the negotiation of work-place cultures in multi-national settings, and then apply these frameworks to the data described in section 4. Finally, in section 6, I discuss the value of analysing software developments from a cultural perspective.

2. A culture of professional end-user development

Culture is a notoriously difficult concept. In this paper, I take the view that the culture of a particular group of software developers consists of a set of common customary behaviours, assumptions and values.

Over the past ten years or so, I have conducted a number of field studies of professional end-user developers, financial mathematicians, earth and planetary scientists and biologists, developing software

either on their own or in partnership with software engineers in order to further their own professional goals (Segal 2001, 2005, 2007). These field studies have formed the basis for an exploration of the culture (values and customary behaviours) of scientists as professional end-user developers (Segal, 2008a, 2008b; Segal & Morris, 2008). They reveal a pervasive model of professional end-user software development as depicted in Figure 1.

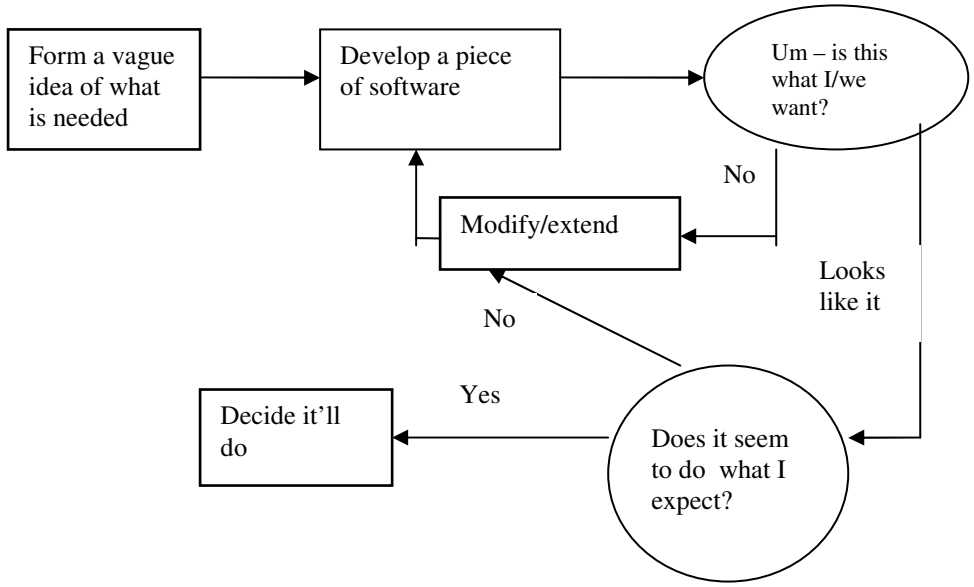

Figure 1:A model of professional end-user software development, adapted from Segal and Morris (2008)

This model is basically an iterative, incremental feedback model, which depends for its efficacy on the developer being embedded in a cohesive user community. In order to begin the process, the developer must have knowledge of the domain and intuition as to how the software might be used. S/he then develops a piece of software, and reflects on whether it does what is required, involving his/her colleagues where necessary. This involvement can be very informal, for example, the developer asks his/her colleague working at the next bench to 'come and have a look at this', or discusses issues with colleagues during coffee breaks. Depending on the feedback received, the system is then modified or extended until the point at which it is thought to fulfil its purpose. It is then tested very cursorily, with testing typically consisting of the scientist inputting some representative data and deciding whether the output looks reasonable.

There are certain assumptions and values underpinning this model:

1. The software is valued only to the extent that it supports the science, see also for example, Sanders and Kelly (2008), Basili et al., (2008).
2. Allied to this is the fact that domain knowledge and skills are valued far above software development knowledge and skills. In Segal (2007), I discuss the pervasiveness of the attitude among professional end-user developers that 'anybody can develop software'.
3. The developer has a deep knowledge of the scientific domain and the requirements. This is necessary both for the start of the development (the developer starts with a high level idea of what is needed) and, perhaps more importantly, for the conclusion (the developer must have a deep knowledge of the domain in order to have the gut instinct that the software 'will do'). One consequence of this is that neither the establishment of requirements nor testing is regarded as a major issue.
4. The developer is either a potential user himself/herself or is embedded in the user community. In either case, the assumption is that users will be an integral part of the development process.

5. The development process is driven by the immediate needs of the user group involved in the development. Development issues associated with the use of the software across time, space, diverse user groups – issues such as code comprehensibility, robustness, modifiability, portability etcetera – are treated lightly, if at all. This comparative disregard of some of these major issues of software engineering together with the lack of recognition of the difficulties of establishing requirements and testing in general as noted in 3 above, though perfectly reasonable within a professional end-user developer context, might contribute to the lack of value ascribed by professional end-user developers to software development knowledge and skill as noted in 2.

Judged by its pervasiveness in contexts where these assumptions hold, the model of development in Figure 1 is very successful. Conversely, the success of the model in Figure 1 depends on the assumptions 3, 4 and 5 above being met. There is the risk, however, that scientists who are accustomed to this model might believe at some level that the values and assumptions underlying it apply in any development context. I shall discuss this risk in more detail in subsequent sections of this paper.

It should be noted that none of my field studies concerned computational scientists, such as climatologists or nuclear physicists, who dedicate their careers to maintaining and developing an extremely large code base over a long period of time using the tools and techniques of high performance computing (HPC). The culture of these scientists does not quite fit with the model above: field studies in HPC settings, see, for example, Carver et al. (2007) and Kelly (2007), show both similarities and differences with my studies. Among the similarities are the emphasis on the science and the necessity of domain knowledge within the development team. The differences include the facts that, given the longevity and complexity of HPC code, these scientists both take cognisance of the problems of maintainability, portability etcetera, and are given due respect for their development knowledge and skill.

The biologists described in this paper are not HPC developers; their experience of developing software is as in Figure 1 and their underlying values and assumptions are as described above.

3 The context of my field study

The software, FSLIMS, at the centre of this field study is a Laboratory Information Management System, a LIMS, for the storage, retrieval, reporting and mining of laboratory results in a particular biological context. In this section, I shall briefly describe the biological practice that FSLIMS is intended to support; the major groups of stakeholders including the development team; the role of collaboration within and between these groups; and, finally, the methodology of this study.

3.1 The biological practice that FSLIMS is intended to support

The biological practice that FSLIMS is intended to support is that which is carried out in (what are commonly referred to as) wet labs where gene sequences are transformed into protein crystals. Having obtained the crystal, there are then established procedures and software packages for inferring the structure of the protein from this crystal. The structure of a protein gives vital information as to the function of the protein and how it binds with other biochemical objects, and this information is very important in, for example, the understanding of diseases and the design of drugs. While the nature of protein crystallography is reasonably well known, in that there is an established body of knowledge and a well established set of software packages for supporting the crystallographer's work, the same is not true for the wet lab work. The transformation of the genomic DNA to the protein crystal is a multi-step process, often referred to as a pipeline, with somewhat different steps for different proteins, and with varyingly high failure rates at each step. Figure 2 below illustrates a typical process which takes as input the genomic DNA as identified by bioinformaticians and outputs the protein in crystalline form. The nature of the exact steps in the figure are not relevant to this paper, and in any case, may vary depending on the type of protein (membrane or enzyme or…)

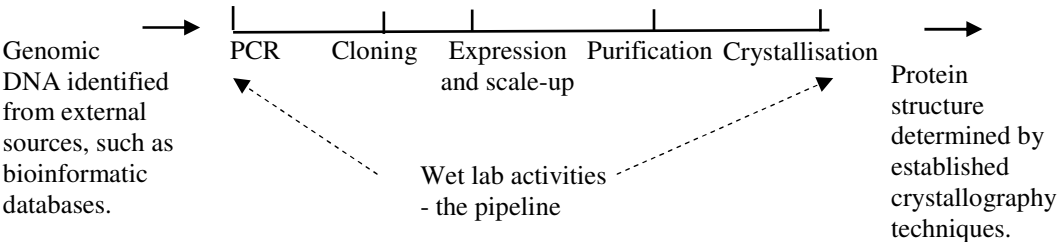

Figure 2. Activities in a wet lab as supported by FSLIMS.

A biologist attempting one of the steps in this transformation (for example, the 'expression and scale-up' step) is entering relatively unknown territory. It is not known which procedures are more likely to lead to success and failures are common.

Borrowing from the terminology of evolution, Knorr Cetina (1999) describes the master strategy that molecular biologists use when faced with a problem in carrying out a procedure as 'blind variation and natural selection' . In evolutionary terms, 'blind variation' means random variation. Knorr Cetina allows that the molecular biologist's choices of variations, in, for example, the temperature or the length of time that one substance is left soaking in another, are not totally random. Rather they are 'semi-blind' in that their choice is dependent on the biologist's experience of empirical reality. 'Natural selection' depends on success. If a procedure produces useful results, it may become part of the armoury of the molecular biologist, become communicated to other biologists in the laboratory, and eventually become part of the community's established procedures.

The procedures for carrying out the transformation steps from genomic DNA to protein crystal are still, at the time of writing this paper, at the stage of 'blind variation and natural selection'. Carrying out a particular procedure in order to achieve one of the transformation steps in Figure 2 is referred to as conducting an experiment by biologists. (This use of the term 'experiment' might be queried by other scientists such as physicists, since no hypotheses are being tested: a better term might be 'trial'. Still, 'experiment' is the term in general usage and the one that will be used in this paper.) These experiments are often carried out in parallel by way of plates, with a plate containing 96 (or, sometimes 4 x 96) wells with each well containing slightly different contents. Thus 96, or 4 x 96, experiments are carried out simultaneously. This use of plate experiments aims both to address the high failure rate of individual experiments and to meet the needs of high-throughput labs. These latter attempt to find the structure of many proteins, as opposed to low-throughput labs which concentrate on a few proteins perceived as being of especial importance. Labs, either working individually or in consortia, might have one set of researchers (or one lab in the consortium) concentrating on one transformation step in Figure 2 and handing on their results to another set (or lab) for the next step.

At the time that FSLIMS was proposed, biologists used lab books to record their procedures and results (so this knowledge could be lost when biologists left a lab and took their lab books with them) or electronic-office-based means unique to particular labs, such as Word templates. In the latter case, data aggregation within a single lab was difficult as was exchange of information between labs (because each lab might use a different template), data mining across the community was impossible, and – following the conventions of scientific publication, when failures are almost never published – the conditions under which an experiment had failed were not recorded. There was thus nothing to stop biologists wasting time on pursuing experiments under conditions which had a history of leading to failure.

It became clear that it was important for the conditions and results of each experiment, whether a success or a failure, to be recorded electronically in a structured form in a LIMS for the following reasons:

1. so that individual scientists have an electronic repository of their work from which they can extract details to include in scientific papers making use of shared libraries of procedures where appropriate.

2. so as to form a lab-based knowledge resource and safeguard against the potential loss of information when individuals leave the lab.
3. so that experiment results can be handed on to other scientists further up the pipeline.
4. so that the heads of labs can have access to aggregated data informing them of the progress of the various experiments in their labs.
5. so that, in the fullness of time when enough data has been recorded, data mining techniques can be performed with the aim of ascertaining the conditions under which a particular experiment is likely to be successful. By this means, successful procedures can be identified and become established in the community.

One challenge facing the FSLIMS developers is to negotiate the minefield of terminology. The molecular biologists in this field study did not always agree on their use of terms: different labs used the same term with different meanings. In fact, the biology community as a whole is used to working within a framework of terminological ambiguity. For example, there is no systematic naming system for proteins so I was told that it is not uncommon for two people to be working on exactly the same protein and calling it different names. But of course this terminological ambiguity poses problems for a community-based LIMS (which is essentially a database). Biologists from different labs might enter data with different meanings into a field identified by a particular term, which makes such data difficult to share and impossible to aggregate across labs. Or the LIMS developers might feel obliged to impose semantics on a biological term, which seems an inappropriate action for developers to take.

3.2 The stakeholders and communication/collaboration.

There are five major groups of stakeholders in this development, as illustrated in Figure 3.

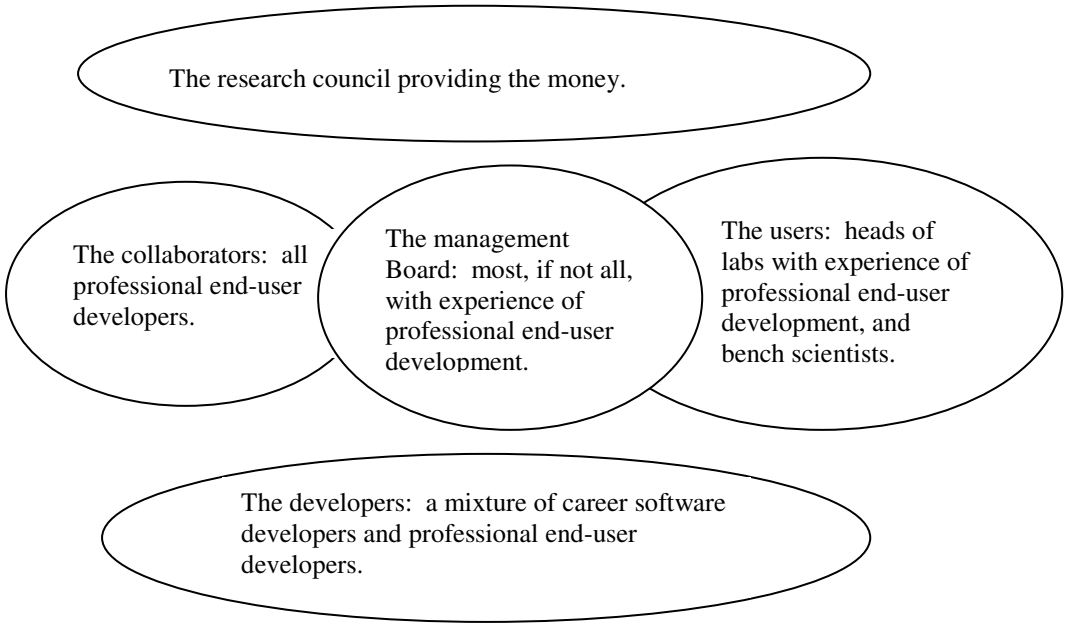

Figure 3: the major stakeholders

A research council representing the wide community of UK biologists is providing the money both directly and indirectly via research labs, and needs to be satisfied that this money is being well spent from the community's point of view. This group of stakeholders is included here for the sake of completeness: it will not be referred to further.

The management board of the project consisted initially of ten people (of whom one has since somewhat disassociated herself from the project) including an overall manager. The board consists of both protein biologists, of whom most, if not all, have had experience of professional end-user development during the course of their careers, and people referred to as 'tool builders'. These latter are professional end-user developers who construct software in order to support wet lab work or to

support the use of bioinformatics databases or crystallography at either end of the wet lab pipeline as illustrated in Figure 2.

Overlapping with the management board are two more stakeholder groups: the collaborators and the potential users of FSLIMS, the biologists in the labs. The collaborators consist of professional end-user developers from research groups within both the UK and the rest of Europe who have the potential to share resources (expertise, source code) with the FSLIMS development team. The user group consists both of the heads of labs, prestigious scientists who are also on the management board and who take advantage of the reporting facilities of FSLIMS, and bench scientists, who run the experiments and enter the data. The initial user group consists of two consortia of labs doing similar work, and two individual labs. It is hoped that this user-base will expand in future.

Finally there is the development team. At full strength, there are nine members of the development team distributed over five locations. There is much diversity in this team: there are seven men and two women; some are co-located with specific groups of users and some are not; some are funded by the FSLIMS project and others by individual labs or consortia; some work full-time on FSLIMS and others have the overall remit of supporting all software in a particular lab or consortium so that working on FSLIMS is only part of their duties; in terms of professional background, four are software engineers, four are professional end-user developers and one is a bioinformatician; they have been brought up and educated variously in France, Russia, China, South Africa and the UK; most have never worked together before. With such a wide diversity and distribution, it is not surprising that tensions between members of the development team quickly became evident and it took much time and effort before the team melded into a coherent whole (see 4.6 below).

Table 1 articulates the role of cooperation between and within the various groups in Figure 3 omitting the research council which is included in Figure 3 solely for the purposes of completeness. The actuality of cooperation is described in section 4 in those parts indicated by italics in column 1.

Cooperation	**Role of cooperation**
Between the development team and the collaborators *(4.2)*	To pool software resources
Within the members of the management board. *(4.3)*	To resolve potential conflicts between high-level requirements and then to prioritise them. To agree terminology (see 3.1).
Between the development team and the management board *(4.4)*	The management board: to communicate the prioritisation of requirements and the agreed terminology, and to check that the system in its current state is meeting, or has the potential of meeting, the scientific goals. The development team: to communicate the resource implications of potential prioritisation decisions, and the current state of development.
Between the development team and the wet lab bench scientists (the people who are entering data into FSLIMS) *(4.5)*	To ensure that the software meets the users' needs; to ensure usability; to gain buy-in from the users.
Within the development team *(4.6)*	Probably a necessity for any successful development

Table 1: The role of cooperation within and between different groups of stakeholders

3.3 Methodology

Data for this ongoing field study is being collected using observations, interviews, phone calls, emails, and project documentation, and is thus strongly triangulated. Where information comes from only one source, I have indicated this herein by the use of expressions such as 'as I was told by x…'.

At the time of writing, there have been ten observations. Six of these have been of day-long face-to-face development team meetings, two have been two-day workshops with the developers, the users and other stakeholders, and two have been of two-day 'code camps', in which most of the development team works in co-location with a user lab. Besides the short interviews (less than half an hour) which I have been able to carry out with various stakeholders during the observations, there have been ten individual interviews of duration from half an hour to two hours which have been audio-taped and fully transcribed. There have been five further substantial face-to-face interviews and twelve phone interviews, which have not been taped but have been fully noted. Those interviewed include:

- the project manager repeatedly;
- all the other members of the development team, some repeatedly;
- all the main collaborators;
- from the management board, the overall manager (repeatedly), the man designated the lead scientific representative and, very briefly, two well-known protein scientists;
- a few users.

The relative lack of contact with users is explained by the fact that this field study focuses on the early development stages of FSLIMS and the number of users at this early stage was small for reasons that will be explained later. I am hoping to focus more on users as the development proceeds.

In addition, there have been numerous emails as I have sought, or been offered, clarification, and the FSLIMS project has a good deal of associated documentation to which I have been given open access.

I am analysing the data by splitting it up into semantic units and then grouping these units into themes using an inductive iterative coding scheme. So far, there are six main themes labelled context, developers, development, requirements, users, and data/data model, and fifty seven sub themes.

As to validating my interpretation of the data, I have given a seminar based on the ideas contained in this paper to the development team and sent an incomplete early version of the paper to the three main sources of information (the project manager, the overall manager, and a representative of the collaborators). All have responded very positively in terms of engaging with the material, and the final paper reflects many of the clarifications, amendments and extensions that they have suggested. The main argument of the paper – that many problems in the early stages of this development may be attributable in part to the influence of the professional end-user developer culture described in section 2 – has been greeted with universal recognition.

4 Problems with cooperation

4.1 Two initial constraints

Before I consider the problems of cooperation which emerged between and within the four major groups of stakeholders identified in Figure 3, I shall describe two constraints imposed on the development at the very beginning: the staffing of the development team and the necessity of producing a quick initial system. Both of these constraints impacted heavily on the early stages of the development.

4.1.1 Staffing the development team

It is widely accepted that, if only because of the complexity of the domain, potential users must be heavily and effectively involved in the development of scientific software, see, for example, Segal and Morris, (2008). Many approaches to software development, such as user-centred and participatory design, focus on user involvement while admitting that effective user engagement is not a trivial issue, see, for example, Kujala (2003) and Wagner and Piccoli (2007). With the burgeoning of e-science, there has recently been much interest in how effective user involvement might be achieved within a scientific context, see, for example, De Roure and Goble (2009), Macaulay et al. (2009) and Thew et al. (2009), as well as Letondal (2005). It thus seems reasonable that, in putting together the development team, consideration should have been given to the team having expertise in enabling user involvement. Such, however, was not the case.

Of the four software engineers, one is the project manager/lead developer who, at the time of his appointment, had considerable experience of working on back-end systems but very little of negotiating with users. Two other software engineers at the time of their appointment were at the beginning of their careers and so had very little experience of any development, and the fourth one only joined the team some way into the project. It might be that, in making these appointments, the biologists and professional end-user developers on the appointment panel were not aware of the necessity of the team having to have expertise in negotiating explicitly with users, since the fact that users are integral to the process of professional end-user development, as described in Section 2, means that user negotiation is not a major issue in this context.

As to appointing the professional end-user developers to the team, the overall manager on the management board told me that, in general and in this specific instance, people are often recruited to develop software not on the basis of their software development skills but because they are perceived as being useful people within their research groups and their current funding is in jeopardy. In fact, two of the professional end-user developers appointed had very little experience of development. This attitude towards development team appointments is totally consistent with the attitude expressed in section 2 as ‘anybody can develop software’.

4.1.2 The immediate necessity for FSLIMS: the initial version

One major constraint on the development of FSLIMS was that some of the labs had an immediate need for it: these labs were already generating the data which it was intended to manage. There was thus no time to develop FSLIMS from scratch: the labs had to be provided with an initial version of the system as quickly as possible. It was hoped that the labs would then provide feedback on this initial version and that development would proceed based on an incremental, iterative feedback model.

The initial version had of necessity to be largely based on components produced by collaborators which were already in place. Underlying FSLIMS was a UML model of protein production. This model had been developed in collaboration with the structural protein community in both Europe and North America. Associated with the model were mechanisms by which developers could write Python scripts for the automatic generation of both a database schema and a Java API. There was also planned to be input (e.g. sharing of code) from collaborators who had already invested effort in developing LIMS for their own labs.

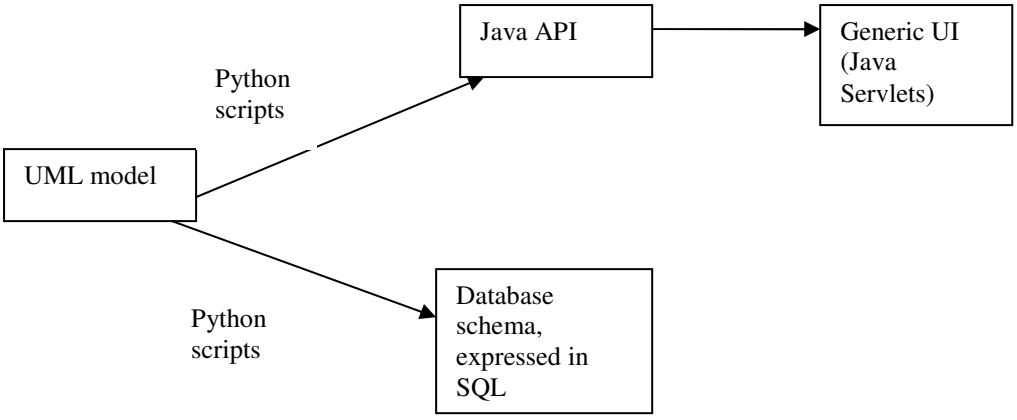

Figure 4. The initial architecture of FSLIMS

The architecture of the initial version, as in Figure 4, was that of the data model with the automatically generated database schema and Java API, and a user interface (UI) developed from this API with a view to providing a window on the whole data model and generated database.

The decision to have an entirely data-driven UI goes against the ethos of the commonly accepted three schema architecture of database design, which essentially separates the concerns of modelling and storing data from the concerns of the data users. The project manager justified this decision as follows:

- the data model was very complex and the UI provided a window by which to understand it.
- At this initial stage, the FSLIMS team had little shared knowledge of lab practice and hence of detailed user needs. Given the fact that the existence of software to support a task often changes the ways in which the task is done, the project manager thought that he could not predict with any certainty how the system would be used, and thus he decided that scientists should be given free rein to explore the software and use it as they wish. He anticipated that usage data logs could then be deployed to identify common usage patterns, and development resources expended to support these.
- constructing a UI from components already in place provided the necessary quick initial fix.

The fact that the UI was data- rather than user-driven was undoubtedly a major factor in the very poor take-up of the initial version. The data model had been developed with the intention of being all-embracing as regards proteins and thus contained many elements which were of no consequence to FSLIMS. Given the automatic generation mechanisms and the data-driven focus of the initial UI, these irrelevant elements could not always be ignored by users and led to a perception that the system was too complex to be usable.

Users continued to record their data using other methods. One consortium continued developing their own LIMS based on the needs of their own users and somewhat lacking in structure; I shall refer to this LIMS as CLIMS ('Consortium Laboratory Information Management System'). One of the labs struggled with a commercial LIMS which was somewhat inadequate for research purposes; other users continued with their existing practices of using Word documents or Excel spreadsheets. This lack of engagement with FSLIMS and the concomitant lack of user feedback meant that the strategy of incremental, iterative development was very difficult to achieve in the early stages.

It became clear that a user-driven focus was necessary. The new starting point was not the data model but CLIMS. It was recognised that CLIMS code could not form the basis of FSLIMS's code – for example, data was not adequately structured - but it did encapsulate the then current requirements of the users in the CLIMS' consortium. The decision was made that CLIMS be used in parallel with FSLIMS until such time as FSLIMS supported the requirements encapsulated in CLIMS, when CLIMS

would be dropped. The head of the CLIMS consortium made it mandatory that FSLIMS/CLIMS and then FSLIMS be used in the consortium's labs.

4.2 Problems of cooperation between the development team and the collaborators

There were essentially two groups of collaborators, all of whom were professional end-user developers and thus steeped in the values, concerns and customary behaviours described above in section 2. Cooperation between them and the FSLIMS project proved to be extremely problematical and, in fact, eventually broke down completely.

One of the groups of collaborators had prime responsibility for maintenance of the protein production UML model which underpinned early versions of FSLIMS as in Figure 4. These data modellers are accustomed to developing software either on their own or within a particular team of professional end-user developers. In the former case, each developer works according to the model of Figure 1 and has autonomy over the development provided that it is perceived that the software adequately supports the science. Where development is a team effort, the model in Figure 1 needs to be extended in order to encompass the negotiations that take place when disagreements arise between members of the team regarding implementation details. These disagreements are accepted as a normal part of working life and the ensuing negotiations are relatively painless. Thus, authority for implementation is invested in the team consensus. Reaching such a consensus is made easier by the fact that the team operates as a community of practice, see, for example, Brown & Duguid (2000). Members of the team are all professional end-user developers working in the same scientific domain and so share the same goals and the same values and customary behaviours as discussed in section 2. They have been working together for a period of years, are working either in the same office or in adjacent offices, and have tacit knowledge of individual strengths and weaknesses and of the distribution of both domain expertise and software development knowledge among the members of the team.

Drawing on their experience, the data modellers expected that implementation authority in the FSLIMS development would be by the consensus of themselves and the project manager. This expectation was confounded by the latter who expected that the authority (and responsibility) for implementation decisions be invested in him. Such an expectation seems entirely reasonable given his greater experience of dealing with software engineering issues such as testing which are of major concern in the FSLIMS context but treated only lightly in professional end-user development, as discussed in section 2. In any case, a key document dating from the inception of the FSLIMS project affirmed this authority. The collaborators appeared to know nothing about this document. I have noted before in Segal (2005) the cursory use that professional end-user developers make of project documents given their reliance on face-to-face communications.

The upshot of this clash between different notions of implementation authority was that technical disagreements between the collaborators and the FSLIMS project manager were seen as confrontations by both sides and communications between the two parties became very difficult. These difficulties were exacerbated by the lack of a community of practice encompassing FSLIMS and the collaborators similar to that which enabled consensus among the collaborators. They led to FSLIMS eventually becoming independent of the original data model.

Clashes over implementation authority also played a major part in the dissolution of the relationship between FSLIMS and another potential collaborator. The latter was a professional end-user developer who was in the process of developing a LIMS for her own lab. This LIMS was underpinned by the protein production model as in Figure 4, but the user interface reflected the user needs in her lab rather than being driven by the model. Unlike the situation with CLIMS, where it was recognised that the value of CLIMS lay in its explication of one consortium's needs and that it would have to be re-implemented in FSLIMS (see 4.1.2), this collaborator argued that FSLIMS should adopt her software architecture. However, a LIMS for a single lab – or for a close-knit community of collaborating labs – does not have to address such issues as balancing community against individual interests, which impacts on, for example, access controls to large databases. Thus, technical decisions which are perfectly acceptable for a software system destined for a single lab, or a small group of labs, may not be acceptable when the user-base is a more diverse community. The project manager decided that this was the case here and rejected the proposed architecture as unsuitable. Like the data modellers, the collaborator found it hard to accept the implementation authority of the project manager and the

collaboration foundered. The dissolution of this relationship meant that an early opportunity for the FSLIMS team to gain an understanding of the detailed needs of one group of potential users was lost.

In summary, the fact is that both potential collaborations came to nothing amid some resentment from all concerned. This fact might be explained by the different notions of implementation authority, which might in turn be explained in terms of a clash between the values held by the professional end-user developers and those of the project manager. The former were steeped in a culture which values scientific, over software development, knowledge and skills, and this perhaps made them resent the project manager's assertion of his authority over the implementation on the basis of his greater software development knowledge and experience. This resentment was perhaps exacerbated by what the collaborators perceived to be the project manager's lack of concern with meeting detailed users' needs, as evidenced by the data-driven first version of FSLIMS, and by the lack of a community of practice encompassing both the collaborators and the project manager.

4.3 Problems of cooperation within the management board

As I indicated in Table 1, cooperation between the members of the management board was necessary in order to identify, negotiate and prioritise requirements, and to agree ambiguous terminology. The plan articulated in an early key project document was for the heads of labs to get together to negotiate their high level set of common requirements and then prioritise them in the light of information from the project manager about the resources he estimated to be necessary to implement each requirement. This plan was not put into effect. The heads of labs acted as individuals. Some engaged only perfunctorily with FSLMS as described in 4.1.2. Others delivered their individual requirements to the project manager. The latter had only a weak understanding of the domain and was thus unable to judge how conflicts between individual requirements should be resolved and how requirements should be prioritised. This problem with negotiating high level requirements threatened at one point to block the development completely. The threat was averted by the appointment of one of the management board as 'the scientific representative' who had the specific responsibility of negotiating requirements and then prioritising them in the light of individual requests from lab heads and resource information from the project manager. This appointment proved very helpful in enabling the development to proceed, but, consistent with the relative lack of value ascribed to software development by scientists, as discussed in section 2 above, and given the many other demands made on the scientific representative, the time allocated to him to fulfil this role appears inadequate for the role's demands.

There are several possible reasons why cooperation within the management board to enable the negotiation and prioritisation of requirements did not happen in the early stages of development. The first, I suggest, is to do with the influence exerted by the culture of professional end-user software development on the management board. As I argue in Segal (2008a), it is entirely reasonable for lab heads to communicate high-level requirements to professional end-user developers and then leave them to sort out detailed requirements and all the details of the implementation. It is plausible that the heads of labs in the case of FSLIMS used the same mode of communication in the deep-rooted belief that the project manager could sort out all the details including the resolution of conflicting requirements and the priority of a particular requirement compared with competing requirements. Perhaps the lab heads hadn't quite taken on board the fact that to do this requires domain knowledge of the type possessed by a professional end-user developer but not by the project manager of FSLIMS.

Other possible reasons are more pragmatic. The first is lack of time: all the heads of labs were very busy on a variety of research projects and found it hard to devote adequate time to address any individual project. It is plausible that projects like FSLIMS which enable the production of scientific results but do not directly produce them, suffer most in this respect. Related to this problem are the lab heads' views of LIMS in general, as perceived by the project manager. He told me of a history of unsuccessful LIMS being developed in-house. Given the initial version of FSLIMS, it seemed likely that this too might be unsuccessful. It is thus reasonable that heads of labs would be loath to spend that scarcest of resources, their time, on something which looked like it would lead to failure. A further pragmatic reason is the fact that the heads of labs are all in competition for the limited pot of money held by the resource providers. It is thus plausible that a particular lab head should expend any spare resources on projects unique to the lab whose success might lead to further funding for that lab, rather than on projects which are for the common good of the community, such as FSLIMS.

4.4 Problems of communication between the development team and the management board

The influence of the professional end-user development culture plausibly also played a part in the tensions between the management board and the FSLSM development team caused by the latter perceiving the former as making inappropriate technical decisions. At one point, the team made a formal complaint to the board about what they perceived as interference in technical matters. This was at a point in the development when the management board had decided that the current implementation could not meet the goal of data mining. The team felt that whereas it is certainly part of the management board's role to understand enough about the implementation to be able to judge whether or not FSLIMS has the potential to meet the scientific goals, it is not part of this role to decree how the implementation might be changed. Apart from the issue of roles, there is the question as to whether the members of the management board have the requisite expertise to make implementation decrees. Given that their experience is of professional end-user development in a context in which the values and assumptions of section 2 all hold, the answer is almost certainly no. In particular, it is unlikely that the management board had experience in addressing issues such as rigorous testing or code maintainability and portability. It is very likely that the board did not recognise this limitation in their expertise.

From the point of view of the management, given the history of unsuccessful LIMS developments (see 4.3 above), they may have had little trust in the development team and felt that intervening in the implementation was both necessary and helpful. As the development progressed, I did not hear of any further interventions, which suggests that the management's trust in, and respect for, the development team's ability increased. There is further evidence for this from the fact that the management board have stated an objective of keeping the development team together for further projects.

4.5 Problems of cooperation between the development team and the wet lab bench scientists

Having considered high level, I now consider more detailed requirements. These needed to be established by the development team in conjunction with the potential users of FSLIMS, who are largely the wet lab scientists working at the bench.

Given the co-incidence of the starts of both the FSLIMS development and the biology project it was specifically intended to support, the original project plan had as a very early milestone the delivery of an initial version of FSLIMS with some real but limited functionality. The project manager at project inception thus valued the rapid delivery of this initial version over meeting the detailed needs of users. As described in 4.1.2, this version was to a large extent unusable. By the time FSLIMS had been developed to the point at which it was theoretically usable enough to engage the users, many of the latter were loath to change any of the alternative data recording systems to which they had become accustomed, especially given that many were a good way into their fixed-term contracts. It is thus entirely reasonable that such users were reluctant to expend any of their energy in engaging with the development of a system they were unlikely to use. Even in those labs where the use of CLIMS/FSLIMS was mandated, I was told by the project manager that users were sometimes loath to tear themselves away from their science to speak to the developers.

The value ascribed by the project manager to meeting the detailed needs of users increased over the course of development to the extent that at the time of my revising this paper, the entire development team is agreed that the biggest risk facing FSLIMS is a potential lack of take-up, and recognises that meeting detailed user needs and addressing usability concerns in general are essential to mitigate this risk.

The development team have encouraged user feedback by organising code camps, in which most of the development team relocate to a user location for a few days, and user workshops. The extent to which these code camps and workshops have been effective is still to be determined, and engaging users meaningfully in the development activities remains a challenge.

4.6 Problems of cooperation within the development team

I began this field study about 18 months after the inception of the project at the invitation of the project manager. He was concerned about what he perceived to be the lack of cohesion of the development team and thought my presence might encourage more team reflection. At this point, the development team was facing the major problems caused by the initial version (4.1.2) and the difficulties of prioritising high level requirements (4.3). The problems arising from the collaboration with the data modellers were still to come (4.2). The latter two adversely affected the cohesion of the team. The lack of progress on prioritising requirements meant that those members of the team funded by labs promoted the interests of those labs without concern for the general good of the community, and the tensions between the data modellers and the project manager weighed heavily on the shoulders of the member of the development team who was co-located with the data modellers and worked most closely with them. As to the heterogeneous composition of the team, as described in 3.2, the distributed nature of the team made discussion (especially of the informal agenda-less type) very difficult. Other factors contributing to the heterogeneity of the team, gender, national background and professional background (professional end-user developer or software engineer), have had no significant discernible influence.

Despite these early difficulties, the common perception is that the team has now gelled: in general, and despite some tensions between people focussing on the user side and people working on the back-end, it is functioning as one cohesive unit. It is not clear how this gelling was affected: perhaps it was due to the team members' increasing familiarity with each other (they come together every couple of months or so). The project manager is of the opinion that weathering the various setbacks at the beginning of the development had a very positive effect on improving the cohesion of the team. In addition, the appointment of the scientific representative in response to the problem of prioritising requirements (see 4.3) resulted in clear goals being set for at least the next year of development. The FSLIMS team thus had a clear community-led focus (as opposed to having to grapple with the requirements of individual labs and consortia), and this clear focus was (in the project manager's opinion) a strong factor in the gelling. On his part, the project manager has arranged sessions on 'soft skills' such as conflict management for the whole team and now encourages more free-ranging debate at team meetings rather than focussing on specific implementation issues from an agreed agenda as was the case at the beginning of the development.

My interviews and less formal questioning of the team members presumably did raise the level of reflection in the team as the project manager had hoped. I gained the impression that the team were all very reflective practitioners and very happy to reflect on the broader issues of their work for my benefit. (This, in fact, has been a common characteristic of all my studies: I invariably find that, *provided* they have the time, participants welcome the opportunity to speak to me.) But whether this had any effect on team cohesion I cannot say.

4.7 Summary: the problems of cooperation

The problems of cooperation described in this section (which are not necessarily independent of each other), the subsequent effects on the development and the resultant changes, are summarised in Table 2.

	Problem description	**Effect of the problem**	**Resultant change**
1	The development team as appointed had no expertise in user-centred development (4.1.1).	Enabling effective user engagement was not initially given a high priority.	The development team became far more aware of the importance of users as the development progressed.
2	The initial version of FSLIMS, data-centred rather than user-centred, had very poor user take-up (4.1.2).	The proposed incremental, iterative feedback development model was very difficult to put into practice.	The implementation changed from being data-centric to being more user-centric.
3	Collaboration in the implementation with professional end-user developers was a failure	There was a total breakdown in the collaboration. From the FSLIMS point of view, this meant that access to some groups of potential	Collaboration with the original groups of collaborators became

	(4.2).	users was lost.	impossible.
4	The prioritisation of high level requirements by the management board acting in concert was not done (4.3).	This lack of high level requirements prioritisation made it very difficult for the development to proceed. In addition, it increased tensions within the development team as members co-located in or funded by different labs promoted the interests of these labs above those of the community.	A scientific representative was appointed and given the responsibility for requirements prioritisation and for liaison between the development team and the management board.
5	Communication between the management board and development team was ineffective (4.4).	The development team felt resentful that, in their view, the management board was interfering in the implementation by making inappropriate technical suggestions while at the same time abrogating their responsibility of prioritising requirements.	As above, the scientific representative improved prioritisation and communication. The board stopped making specific implementation suggestions.
6	Involving users in the development was difficult (4.5). This was exacerbated by problems 1 and 2 above.	Lack of effective user involvement made it difficult to execute an incremental, iterative feedback development model, and to ensure that FSLIMS met the users' needs.	As 1. The developers organised code camps and workshops in an effort to improve user engagement.
7	The development team was distributed and some had divided loyalties between individual labs/consortia and the community (4.6).	Members of the team co-located with a particular user lab urged the interests of that lab above those of the community, see 4 above. The distributed nature of the team made discussion and debate difficult.	The improvement in prioritising requirements gave the team a clear vision for proceeding with the development in terms of community goals. Wide-ranging discussion and debate, as opposed to the addressing of very specific implementation issues, was encouraged at team meetings. 'Soft skills' training sessions were held for the development team at the instigation of the project manager.

Table 2: A summary of the problems of cooperation and the resultant changes in the development context

In the next section, I shall analyse these problems from a cultural perspective.

5. Software development cultures and cooperation problems

I am not aware of any studies, other than the one described in this paper, which analyse cooperation problems within a software development in terms of different software development cultures. With the burgeoning interest in globalisation, however, there are several studies in the literature which analyse cooperation difficulties in terms of the different *national* cultures of the participants. I see no reason why the various lenses brought to bear on the latter context might not also be trained on the former.

Thus, in this section, I shall firstly describe some relevant literature on multi-national work cultures, and then apply some of the analytical frameworks used therein to the case of the FSLIMS development.

5.1. The culture of the workplace and national cultures: some relevant literature

There are various lenses through which one might analyse culture in the context of the workplace. One of the best known of these is that of Hofstede, see for example, Hofstede (1994), who identified four (later, five) dimensions by which national cultures might be analysed and compared, and provided categorisations of different national cultures according to these dimensions. His aspiration was that these categorisations be used to identify and manage problems which arise within multi-national work teams. However, it is generally acknowledged that Hofstede's work has its limitations, see, for example, the discussion in Brannen and Salk (2000) and in Walsham (2002). For example, Hofstede pays no cognisance either to the dynamic nature of work cultures or to the heterogeneity within cultural groupings. In addition, as Walsham points out, Hofstede's national categorisations can be difficult to apply to detailed work patterns.

Whilst admitting that any particular analytic framework will emphasise some aspects of culture at the expense of others, Walsham suggests that structuration theory provides a stronger tool than Hofstede's categorisations in considering activities in the workplace. Structuration theory regards structure and activity to be a duality, in that structure guides actions which in their turn produce or reproduce structure. This duality is thus essentially dynamic (as is workplace culture). It can be analysed along three interlinked dimensions, systems of meanings, norms, and power relationships, which Walsham claims have more immediate relevance to work patterns than those of Hofstede. Culture is conceptualised as an overall systematicity of these three dimensions within a social collectivity but individual variations are recognised. Conflict is not an inevitable consequence of cross-cultural contradictions but rather occurs when such contradictions are present, when their impact on certain actors is perceived to be negative and when the actors are able to react to this perception.

Brannen and Salk (2000) also focus on work cultures as dynamic entities. By means of a case study, they illustrate how people from different national cultures might negotiate a viable culture of work fashioned by such contextual characteristics as emergent threats and the stances of the individuals involved with respect to a particular cultural norm of their group (thus acknowledging individual variations within cultures). The emergent negotiated culture might involve elements where one group has compromised its particular cultural norm (so that the norm of the other group prevails); elements where both groups have found some middle ground (so that the norms of both groups with respect to this element are modified); or elements where new norms emerge which were not present in either original culture.

5.2 Cultures of software development and FSLIMS

In this section I draw on the literature discussed above in order to analyse the FSLIMS development as a dynamic work culture in which clashing initial meanings, values, norms and power relationships, are either modified so as to enable the development to proceed or lead to breakdown.

In section 2 above, I articulated, in terms of behaviour and values/assumptions, a (static) culture of professional end-user development in which the FSLIMS collaborators and most, if not all, of the management board were steeped. I did not do the same for the development team as I doubt that, initially at least, the nine members of the team shared many common behaviours, values or assumptions with respect to software development. As described in sections 3.2 and 4.1.1, the team at this time was a very disparate set of career software developers and biologists, two-thirds of whom had either very little software development experience or only a part-time commitment to FSLIMS. One consequence of this is that the project manager dominated the development team, as befitted his experience and as was affirmed in the project documentation. In this section, I thus focus on the incongruences between the values and assumptions of the professional end-user development culture and those of the project manager. The latter differed from professional end-user developers in at least two important aspects:

- At the inception of FSLIMS, he valued reliability and meeting scientific need (that is, high level requirements) over meeting users' needs (that is, the operalisation of the high level requirements);
- He perceived the equal value of both scientific and software development knowledge and skills rather than valuing the former over the latter.

Table 3 revisits some of the problems articulated in Table 2, analysing them from the perspective of incongruent values and assumptions. Table 3 also indicates how the work culture changed in response to these problems. Problems 6 and 7 of Table 2 are omitted from Table 3 as they are, in part at least, knock-on effects of other problems.

	Problem description	**The origins of the problem from a cultural perspective**	**How the work culture changed**
1	The development team as appointed had no expertise in user-centred development (4.1.1).	Appointments were made by professional end-user developers who shared the assumption that 'anybody can develop software'. In addition, it is likely that the need for expertise in user-centred negotiation may not have been recognised by the appointers since in professional end-user developments, the users form a coherent group and are integral to the development.	
2	The initial version of FSLIMS, data-centred rather than user-centred, had very poor take-up. (4.1.2).	The project manager initially prioritised code-focused values (reliability etcetera) over a focus on detailed user needs.	The project manager, along with the rest of the development team, came to appreciate the value of attending to the details of users' needs.
3	Collaboration in the implementation with professional end-user developers was a failure (4.2).	There was an incongruence between the respective values ascribed by professional end-user developers and the project manager to software development knowledge and skills. The professional end-user developers did not value them highly and hence did not give the project manager the implementation authority he thought he was due.	This cultural contradiction led to conflict and the collaboration came to an end.
4	The prioritisation of high level requirements by the management board acting in concert was not done (4.3).	There was a cultural contradiction with respect to the role of developers. In professional end-user development, the developer knows enough of the domain and the user group that he or she is able to prioritise high level requirements; the same is not true of software engineers.	The management board recognised that their understanding of the role of the developer was incorrect in this context and appointed a scientific representative.
5	Communication between the management board and development team was ineffective. (4.4)	As 4. with regards to the prioritisation of requirements. In addition, interventions by the board on implementation matters, deemed inappropriate by the development team, might reflect the lack of value ascribed	As 4 with regards to the prioritisation of requirements. Over time and with subsequent versions of FSLIMS, the

		by professional end-user developers to software, as opposed to scientific, knowledge and skill.	management board grew to trust in, and hence respect, the development team's knowledge and skill.

Table 3: Problems of cooperation and cultural contradictions

Table 3 illustrates some possible causes of the cooperation problems described herein in terms of incongruences between professional end-user developers (the management board and the collaborators) and the project manager: of values (scientific versus software development knowledge and skill, meeting users' detailed needs versus more code-focussed values); of meanings (of the role of developers); and of power relationships (implementation authority). It also illustrates the development of a negotiated work-place culture in which both members of the management board and the project manager modified their initial values so as to enable the development to proceed.

6. Summary and conclusion

In this paper, I describe the problems of cooperation seen in the early stages of the development of a software system intended to support a community of biologists. Informed by some recent literature on the dynamics of multi-national work cultures, I analyse these problems from the perspective of contradictions between the values and assumptions of, on the one hand, the management and the collaborators steeped in a particular culture of software development, a professional end-user development culture, and, on the other, the project manager leading the development team. I describe the dynamics of the workplace culture, the necessary modifications in these values and assumptions, which enabled the development to move forward.

Of course, bringing any sort of analytic lens to bear on field study data will emphasise some aspects of the data at the expense of others. In this study, problems are seldom clearly attributable to a single cause. For example, in 4.3 above, I give several plausible explanations for the management board's reluctance to prioritise high level requirements. Some of those which I label 'pragmatic', for example, those that might be attributable to competition between the heads of individual labs, might be explained in terms of the culture, the norms and power relationships, of research biologists. However, I argue that the lens of software development culture is an especially appropriate one for studies of scientific software development in the light of the pervasiveness of the model of professional end-user development culture that I describe in section 2 herein. Support for this argument is provided by the responses I receive when I talk about this model to both professional end-user developers and software engineers. As I mentioned in 3.3, the response of the participants in this study to the notion that many cooperation problems in the FSLIMS development were due to the influence of the culture of professional end-user development, was universal recognition that yes, this is so. And whenever I discuss the model with software engineers who know nothing of FSLIMS but who have experience of developing scientific software, they respond with the same recognition. I am thus led to the inescapable conclusion that both researchers and participants in scientific software developments should take cognisance of the culture of professional end-user software development, the former because of the extent of its influence as discussed herein, and the latter in order to anticipate, identify and possibly alleviate cooperation problems.

My final point is this: I do not think that professional end-user developers are the only group to have an identifiable culture of software development. For example, Petre and Blackwell (2007) discuss children's unwitting experience of end-user programming in terms of shared behaviours, values and assumptions, that is, in terms of a development culture. I suggest that the whole topic of identifying and examining the effects of different software development cultures presents a rich seam for future research.

Acknowledgements

I should like to express my heartfelt gratitude to those people who participated in this field study. For reasons of confidentiality I cannot name them, but they know who they are. In addition, I should like to thank my colleagues in the Empirical Studies of Software Development Group in the Centre for Research in Computing at the Open University, Marian Petre, Hugh Robinson and Helen Sharp, for their unwavering support of my work. I should also like to thank the anonymous reviewers of an earlier version of this paper for their suggestions and references.

References

Basili, V.R, Carver, J., Cruzes, D., Hochstein, L., Hollingsworth, J.K., Shull, F., Zelkowitz, M. V., (2008): Understanding the high performance computing community: a software engineers' perspective. *IEEE Software*, vol. 25, no.4, pp. 29-36.

Brannen M.Y. and Salk J. (2000): 'Partnering across borders: Negotiating organisational culture in a German-Japanese joint venture', *Human Relations*, vol. 53, pp 451 – 487

Brown JS and Duguid P (2000): *The social life of information*. Boston, Mass: Harvard Business School Press.

Carver J.C., Kendall R.P., Squires S.E., Post D.E. (2007): Software Development Environments for scientific and engineering software: a series of case studies. *Proc. ICSE 2007*

De Roure, D., Goble, C., (2009): Software design for empowering scientists. *IEEE Software*, 26(1) pp 88-95.

Hofstede, G. (1994): *Cultures and Organisations: Intercultural Cooperation and its Importance for Survival*, HarperCollins.

Kelly, D.F. (2007): A software chasm: software engineering and scientific computing. *IEEE Software*, vol. 24, no.6, pp. 120-199.

Knorr Cetina, K (1999): *Epistemic Cultures: How the Sciences Make Knowledge*. Harvard University Press

Kujala S. (2003): User Involvement: a review of the benefits and challenges. *Behaviour and Information Technology*, vol. 22, no.1, pp. 1-16.

Letondal C. (2005): Participatory programming: developing programmable bioinformatic tools for end-users. In Lieberman H, Paterno F, Wulf V (eds.). *End User Development*. Springer, pp. 207-242.

Macaulay, C., Sloan, D.., Jiang, X., Forbes, P., Loynton, S., Swedlow, J.R., Gregor, P. (2009): Usability and User-Centred Design in Scientific Software Development. *IEEE Software 26(1)*, pp. 96-102.

Petre, M. and Blackwell, A.F. (2007): Children as unwitting end-user programmers. *VLHCC, IEEE Symposium on Visual Languages and Human-Centric Computing, 2007*, pp. 239-242.

Sanders, R., Kelly, D. (2008): Scientific software: where's the risk and how do scientists deal with it? *IEEE Software*, vol. 25, no. 4, pp. 21-28.

Segal J. (2001): Organisational Learning and Software Process Improvement: A Case Study. In K-D Althoff, R.L. Feldmann, W. Muller (Eds.): *Advances in Learning Software Organizations*, *Lecture Notes in Computer Science, Vol. 2176*. Springer, pp. 68-82.

Segal J. (2005): When software engineers met research scientists: a case study. *Empirical Software Engineering,* vol. 10, pp. 517-536

Segal J. (2007): Some problems of professional end user developers. *VLHCC, IEEE Symposium on Visual Languages and Human-Centric Computing, 2007*, pp. 111-118

Segal J. (2008a): 'Models of scientific software development. *SECSE 08, Workshop on Software Engineering in Computational Science and Engineering*, workshop co-located with ICSE 08, Leipzig, Germany. http://www.cse.msstate.edu/~SECSE08/Papers/Segal.pdf

Segal J. (2008b): Scientists and software engineers: a tale of two cultures. To appear in the 20th Psychology of Programming Interest Group Workshop

Segal, J. and Morris, C. (2008): Developing scientific software. IEEE Software, vol. 25, no. 4, pp. 18-20

Thew, S., Sutcliffe, A., Procter, R., De Bruijn, McNaught, J., Venters, C., Buchan I, (2009): 'Requirements engineering for e-science: experiences in epidemiology', *IEEE Software*, 26(1), pp. 80-87

Wagner E.L and Piccoli G. (2007): Moving beyond user participation to achieve successful IS design. *Comm ACM*, vol. 50, no. 12, pp. 51-55

Walsham, G. (2002): 'Cross-cultural software production and use: a structurational analysis', *MIS Quarterly*, 26(4), pp 359-380